НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Система тестування знань на архітектурі клієнт-сервер»

Виконав (-ла):
студент (-ка) IV курсу, групи ІТ-51
Лемешко Борис Олександрович

_____

Керівник:
Доцент Катін П.Ю.

_____

Рецензент:
_____

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)
Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

ЗАВДАННЯ
на дипломний проект студенту
Лемешко Борису Олекснdровичу

1.Тема проекту «Система тестування знань на архітектурі клієнт сервер», керівник проекту доцент Катін Павло Юрійович, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту

Програмне забезпечення на архітектурі клієнт-сервер. Мови програмування Python та Javascript, середовище програмування PyCharm 2019, обрана для розробки технологія – Django, СУБД –PostgreSQL.

4. Зміст пояснювальної записки

1. Вступ 2. Аналіз вимог до програмного забезпечення 3. Вибір та аналіз технологій розробки 4. Реалізація програмного забезпечення 5. Тестування програмного забезпечення 6. Впровадження програмного забезпечення.

Додатки:
7. Код програми

5. Перелік графічного матеріалу

Діаграма компонентів, діаграма варіантів використання, діаграма послідовності, діаграма бази даних

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вибір тематичного напрямку та узгодження теми дипломного проекту	22.02.2019	
2	Аналіз теоретичних матеріалів та вивчення предметної області	15.04.2019	
3	Розробка технічного завдання, вибір методів та засобів реалізації задачі	24.04.2019	
4	Огляд існуючих рішень з тематики роботи	27.04.2019	
5	Розробка структури прототипу та проектування системи	06.05.2019	
6	Реалізація проекту	20.05.2019	
7	Налагодження та перевірка програми	23.05.2019	
8	Оформлення пояснювальної записки	03.06.2019	
9	Передзахист дипломного проекту	04.06.2019	
10	Доопрацювання пояснювальної записки та підготовка презентації	18.06.2019	
11	Захист дипломного проекту	20.06.2019	

Студент Б.О. Лемешко

Керівник проекту П.Ю. Катін

АНОТАЦІЯ

Лемешко Б.О. Система тестування знань на архітектурі клієнт сервер. НТУУ «КПІ ім. Ігоря Сікорського», Київ, 2019.

Ключові слова: клієнт-серверна архітектура, програмна інфраструктура, діаграма бази даних, діаграма компонентів, оцінювання (тест), Python, Django.

Основна частина документу викладена у пояснювальній записці, виконаній на 69 сторінках, та містить 45 таблиць. Також до його змісту входить 1 додаток.

Приведений огляд існуючих компонентів та технологій, за допомогою яких яких можна провести розробку програмного забезпечення системи тестування знань. Було створено програмне забезпечення на основі мови клієнт-серверного фреймворку Django для мови програмування Python та СКБД – PostgreSQL.

Розроблена система побудована, беручи до уваги вимоги безпеки використання. Була розглянута і теоретично розроблена система збереження даних, визначені основні потоки даних та сценарії їх обробки.

SUMMARY

Lemeshko B.O. " Knowledge assessment system using client-server architecture". NTUU «Igor Sikorsky KPI», Kyiv, 2019.

Keywords: client-server architecture, software infrastructure, database diagram, component diagram, evaluation (test), Python, Django. The bulk of the document is presented in an explanatory note, executed on 69 pages, and contains 45 tables. Also included in its content is 1 application.

An overview of existing components and technologies, through which one is possible to develop knowledge testing system software. Th created software is based on the Django client-server framework for the Python programming language and the DMS - PostgreSQL. The developed system was build taking into account the safety requirements of use. The system of data storage was considered and theoretically developed, the main data streams and their processing scenarios were determined.

Номер рядка	Формат	Позначення	Найменування	Кіл. листів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІТ51.350БАК.002 ПЗ	Пояснювальна записка			
6						
7	А3	ІТ51.350БАК.003 Д1	Діаграма варіантів використання	1		
8						
9	А3	ІТ51.350БАК.004 Д2	Діаграма компонентів	1		
10						
11	А3	ІТ51.350БАК.005 Д3	Діаграма структури бази даних	1		
12						
13	А3	ІТ51.350БАК.006 Д4	Діаграма послідовності	1		
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						

					ІТ51.35.0БАК.001 ТП		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Лемешко Б.О.			Система тестуання знань на архітектурі клієнт-сервер. Відомість технічного проекту	Літ.	Арк. 1 / Акрушів 1
Перевір.		Катін П. Ю.					
Реценз.						КПІ ім. Ігоря Сікорського ФІОТ, гр. ІТ-51	
Н. Контр.		Шинкевич М.К.					
Затверд.							

Пояснювальна записка
до дипломного проекту
на тему: «Система тестування знань на архітектурі клієнт-сервер»

Київ – 2019 рік

ЗМІСТ

ВСТУП

Інформаційні науки в сучасному світі є невіднятною частиною життя людини і охоплюють практично всі сфери життя людини. Серед таких є і сфера освіти. ІТ технології дають змогу більш досконало подати, закріпити теоретичний та практичний матеріал, за допомогою сучасних рішень.

Нині інформаційні технології в повному обсязі використовуються у сфері освіти на різних її прошарках. В школах використовують електронний щоденник – метод звіту поточних оцінок для педагогічного складу школи та батьків дітей, що навчаються в даному навчальному закладі.

В університетах та інститутах також ввели подібну систему. Студенти мають змогу дізнатися свої результати з модульних іконтрольних та атестаційних робіт. Також, в університетах (інститутах) та в школах в повну обсязі використовують онлайн-варіант розкладу занять – для зручності надання учням та студентам актуальної інформації щодо поточного розкладу занять в їхніх навчальних закладах.

В основному, в вищих навчальних закладах та школах використовують паперову систему оцінювання знань. По-перше, паперова система оцінювання недосконала тим, що є можливість нечесного написання (списування), суб'єктивізму при оцінюванні знань, відсутністю електронного обліку та швидкого доступу до інформації. По-друге, у світі, де с кожним днем технічний прогрес сягає все нових і нових висот – використання паперового оцінювання є нераціональним та має негативний вплив на навколишнє середовище.

Метою дипломного проекту є розв'язання проблем несправедливої оцінки здобутих учнями/студентами теоретичних та/або практичних знань. Задля уникнення випадків нечесного та суб'єктивного оцінювання проведених робіт.

Розробка гнучкої системи створення/редагування оцінювань (тестів), інтуїтивного та простого web-інтерфейсу та систему обліку пройдених тестувань (оцінювань) для конкретних навчальних закладів. Компонентами системи є web-інтерфейс, панель адміністратора та серверний застосунок. Результатом має бути

єдина система, яка може надати змогу викладачам та студентам навчальних закладів України (а далі і світу) створювати та проходити тестування з відповідних предметів та певних галузей.

Тест – це певний набір питань з конкретної теми, де кожне питання має визначену кількість варіантів відповідей.

Система надає змогу викладачам стисло та швидко провести, так званий «зріз знань» (тест) з відповідної теми, не витрачаючи додаткового часу на передання білетів з тестами учням або студентам.

Повинна буди впроваджена надійна та безпечна система авторизації для учнів/студентів та вчителів/викладачів навчальних закладів з розгалуженим функціоналом та областю впливу (наприклад вчитель/викладач має дозвіл на створення/редагування оцінювань, а учень/студент - ні).

Сама система не повинна вимагати від користувача додаткових налаштувань, а напрочуд – бути автоматизованою з середини та просити від користувача лише мінімальний об’єм даних для ідентифікації користувача.

Також, система не повинна надавати право вносити данні про навчальний заклад всім користувачам і кожна інстанція (школа, коледж, ВНЗ) повинні подати заявку на реєстрацію в спеціально відведеному місці у web-інтерфейсі. Після перевірки даних про інстанцію, їй надається право на реєстрацію у системі. Далі інстанція (її представник) має заповнити інформацію про факультети/інститути/ направлення (так званий, розподіл) навчально закладу. Після цього вчителі (викладачі) та учні/студенти можуть зареєструватися за даним навчальним закладом та створювати нові тести (оцінювання).

Бакалаврська робота складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 18 найменувань та одного додатку. Графічна частина складається з 4 креслеників формату А3. Загальний обсяг 71 сторінок.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Основною одиницею в комплексному оцінюванні знань – є тест. Задана кількість питань з певною кількістю відповідей – є візитною карткою даного метода оцінки здобутих студентами (учнями) знань. Дуже низький об’єм часу на перевірку написаних робіт – дав тестам велику популярність навіть при паперовому оцінюванні знань.

Якщо більш детально вивчати тему оцінювання знань, можна виділити три основні типи оцінювання:

- письмове оцінювання (відкрита відповідь) – той, хто приймає участь у даному оцінюванні має надати повну відкриту відповідь на поставленні запитання виключно у письмовій формі;
- усне опитування – той, хто приймає участь у даному оцінюванні має доповіси відповідь на поставлене запитання виключно у вигляді усної доповіді;
- тестування – той, хто приймає участь у даному оцінюванні, має вибрати один з доступних варіантів для певної відповіді.

Далі розглянуто кожен тип більш детально.

1.2 Аналіз предметної області

Для ефективного моніторингу досягнень студентів важливо не тільки визначити, що вони знають і можуть, але і об'єктивно оцінити їх знання та навички. Результатом оцінювання повинна бути оцінка, яка включає порівняння того, що вивчають учні, щоб вони могли вчитися у відповідності до вимог навчальної програми.

Користуючись засобами масової інформації та проаналізувавши предметну область, можна визначити, що таке оцінювання. Це - кількісний показник якості результатів навчально-пізнавальної діяльності учнів (студентів).

Письмове опитування. В його основі лежить письмова доповідь з заданої теми. Людина, яка приймає участь в даному оцінюванні, має повноцінно розкрити тему доповіді, дотримуючись певної, визначеної структури. Доповідь оцінюється спеціально відібраною комісією або вчителем (викладачем). В основному проводиться в паперовому вигляді і має велику суб'єктивність при оцінюванні (особисті вподобання або прихильність до певного студента/учня).

Усне опитування. Доповіді (відповіді), що проводяться в усній формі мають біль досконалу форму для перевірки засвоєних учнями (студентами) знань. В основному опитування проводиться «на місці», одразу після одержання теми або доповідачеві (учню/студенту) надається певний час на підготовку д доповіді. Також має суттєву недосконаліть, як і письмове опитування, а саме – можливість суб'єктивної оцінки.

Тестування. Нині є найбільш досконалою формою проведення оцінювання знань серед учнів, студентів та фахівців. Тому приклад – зовнішнє незалежне оцінювання (ЗНО), екзаменаційні білети якого представлені у тестовому вигляді. В основному тести проводять в паперовій формі, тому також є невелика можливість суб'єктивної оцінки – спеціаліст, що перевіряє роботу може навмисно виправити відповіді людини, що оцінюється.

Кожен з наведених типів має високу можливість нечесної та суб'єктивної оцінки. Також, існує можливість, так званого списування – людина, що оцінюється має змогу підглядати до заборонених на оцінюваннях джерел, щоб покращити свою доповідь (іншими словами – списує).

Нині, за допомогою інформаційних технологій та технологій розробки можливо досягти максимальної чесності та об'єктивності оцінювання здобутих студентами/учнями/фахівцями знань.

1.3 Аналіз існуючих рішень

На теперішній час існує багатопрограмних рішень, щодо вирішення проблем оцінювання знань [1-18]. Для побудови досконалої та злагодженої

системи, необхідно ознайомитись з уже існуючими рішеннями. Існує певний ряд рішень, які частково, так чи інакше вже мають реалізацію тих чи інших частин завдання. Нижче приведені існуючі рішення:

1.ЗНО Онлайн

Web-застосунок (сайт) за допомогою якого можна пройти тести наближені до тестів зовнішнього незалежного оцінювання (далі ЗНО). Створено для перевірки абітурієнтами здобутих знань. Форма тестів – наближена до форми білетів ЗНО. Система є доволі простою і інтерфейс не має значних недоліків. Напрочуд – він легкий та інтуїтивний. Пройти випробування може навіть незареєстрований користувач (звичайний відвідувач сторінки). Система має дуже зручний інтерфейс самого тестування, та на нього можна рівнятися, як на приклад для побудови власної системи.

З одного боку – ресурс є дуже популярним, тому що кожного року в Україні дуже великий наплив абітурієнтів і сайт має витримувати високі навантаження. З іншого боку – система обмежена тільки зовнішнім незалежним тестуванням і тому розв’язує завдання неповноцінно. Із сильних сторін варто відмітити зручний і інтуїтивний інтерфейс та гарно спроектовану систему, що витримує великі навантаження. Однак, повного розв’язання проблеми система не має.

2. Online Test Pad

Потужний онлайн-застосунок, який можна вважати найбільшим конкурентом серед усіх існуючих рішень. Система дає змогу проходити та створювати власні тести, опитування, кросворди та логічні ігри. Портал має велику кількість користувачів та створених тестів, опитувань, тощо. Має досить інтуїтивний та простий інтерфейс. Із недоліків – недосконалий дизайн (підбір кольорових рішень) та неповна локалізація українською мовою.

Система вирішує більшу частину поставленого завдання – створення/проходження тестів різних тематик. Однак, вона не надає змогу реєстрування навчального закладу для створення сертифікованих цим закладом

тестувань (оцінювань). Також, треба відзначити, що система надає право на створення тестів, опитувань, кросвордів, логічних задач та інших сутностей тільки зареєстрованим користувачам. Однак, кожен відвідувач сайту, має право на проходження тесту, опитування, логічної задачі, кросворду, тощо.

3. Classmarker

Дуже потужний гігант в області створення та проходження тестів. Основними особливостями є:

- Безпечність та приватність;
- Легкість у визначенні налаштувань тесту;
- Не потрібно встановлювати програмне забезпечення;
- Користувальницькі сертифікати та брендинг іспиту;
- Іспит (тест) можно зробити приватним або доступним усьому світу;
- Розумний помічник, який допомогає керувати обліковим записом;
- Результати автоматично оцінюються та переглядаються в режимі реального часу;

Онлайн сервіс для тестування ClassMarker забезпечує один з найкращих інструментів для створення тестів (оцінювань) для викладачів та підприємств. Використовується в усьому світі для бізнес-тренінгів та тестів, тестів попередньої зайнятості, онлайн-сертифікатів, вікторин з підбору персоналу, охорони здоров'я та безпеки, шкіл, університетів, дистанційного навчання, лідогенерації, онлайн-курсів, електронного навчання, практичних тестів та багато іншого.

Веб-інструмент тестування (оцінювання) дозволяє легко створювати безпечні онлайн-іспити з розширеними налаштуваннями, такими як часові обмеження, публічний і приватний тестовий доступ, випадкові запитання, миттєвий зворотній зв'язок, тощо.

1.4 Аналіз вимог до програмного забезпечення

Щоб визначити вимоги до програмного забезпечення необхідно чітко визначити ролі користувачів в системі. Для повного функціонування системи, користувачі матимуть наступні ролі:

- адміністратор системи;
- авторизований користувач-учень/студент;
- авторизований користувач-вчитель/викладач;
- авторизований користувач-представник навчального закладу;
- неавторизований користувач.

Адміністратор системи – має доступ до панелі адміністрації всієї системи та несе відповідальність за реєстрацію навчальних закладів у системі. Також є головним помічником у вирішені проблем інших користувачів. Має змогу змінювати наповнення (тексти, зображення) web-інтерфейсу системи.

Користувач-учень/студент – після успішної реєстрації та створення профілю, має змогу проходити оцінювання (тести), до яких надав доступ викладач навчального закладу. Має змогу переглядати власні результати з пройдених оцінювань (тестів).

Користувач-вчитель/викладач – після успішної реєстрації, створення профілю та його підтвердження, має змогу створювати оцінювання (тестування) для власної кафедри (якщо тип профілю викладач) або власної школи (якщо тип профілю - вчитель). Також може переглядати статистику здач по своїм оцінюванням та/або оцінювань власної кафедри (школи). Користувач, що підтверджує профілі – представник навчального закладу.

Користувач-представник навчального закладу - після успішної реєстрації та створення власного профілю, має змогу оформити заявку на реєстрацію навчального закладу у системі. Є повноправним адміністратором навчального закладу. Після підтвердження адміністратором системи заявки про реєстрацію навчального закладу, має заповнити інформацію про заклад (додати факультети

та кафедри, якщо навчальний заклад – це університет або інститут). Також підтверджує профілі викладачів, при їх реєстрації.

Неавторизований користувач має доступ лише до інформації, що описує систему та до модулю аутентифікації (реєстрація та авторизація).

Загалом, система має повинна мати наступний функціонал:

- реєстрація користувачів (web-інтерфейс та серверна частина);
- авторизація користувачів (web-інтерфейс та серверна частина);
- створення та налаштування профілю користувача (*учень/студент*, *вчитель/викладач*, *представник навчального закладу*) та редагування профілю (web-інтерфейс та серверна частина);
- створення та підтвердження заявки на реєстрацію навчального закладу. Заявка створюється представником навчального закладу, а підтверджується – адміністратором системи(web-інтерфейс та серверна частина);
- перегляд існуючих оцінювань (тестів) для учнів/студентів та вчителів/викладачів (web-інтерфейс та серверна частина);
- проходження оцінювань (тестів) учнями/студентами (web-інтерфейс та серверна частина);
- збереження та відображення результатів пройденого оцінювання (тесту) для учнів/студентів (web-інтерфейс та серверна частина);
  - створення оцінювань (тестів) вчителями/викладачами (web-інтерфейс та серверна частина);
  - редагування та видалення оцінювань (тестів) вчителями/викладачами (web-інтерфейс та серверна частина);
  - збереження та відображення статистики здач (результатів оцінювань) для вчителів/викладачів (web-інтерфейс та серверна частина);
  - редагування інформації про навальний заклад. Редагується представником навчального закладу (web-інтерфейс та серверна частина);

– створення, редагування та видалення структури навчального закладу. Створення факультетів та кафедр у випадку, якщо навчальний заклад – університет. Структура створюється, редагується або видаляється представником навчального закладу (web-інтерфейс та серверна частина);
– редагування вмісту web-інтерфейсу (тексти, зображення) адміністратором системи (web-інтерфейс та серверна частина).

1.4.1 Розроблення функціональних вимог

Системою, у таблиці 1.1 «Таблиця варіантів використання» передбачено такі варіанти використання:

Таблиця 1.11 – Таблиця варіантів використання

Номер	Назва
UC001	Реєстрація користувача.
UC002	Вибір типу профіля учень/студент.
UC003	Вибір типу профіля вчитель/викладач.
UC004	Користувача направляє на сторінку створення профіля вчителя/викладача.
UC005	Створення профілю учня/студента.
UC006	Створення профілю вчителя/викладача
UC007	Створення профілю представника навчального закладу.
UC008	Авторизація адміністратора системи.
UC009	Авторизація користувача.
UC010	Створення заявки про реєстрацію навчального закладу.
UC011	Проходження оцінювання (тесту) учнем/студентом
UC012	Створення оцінювання (тесту).
UC013	Перегляд статистики.

Продовження таблиці 1.11 – Таблиця варіантів використання

UC014	Огляд власних результатів.
UC015	Редагування профілю користувача.
UC016	Редагування профілю навчального закладу.
UC017	Вихід із системи.

Далі в наведених таблицях 1.2 – 1.18 кожен варіант використання розглянуто детально.

Таблиця 1.2 – Варіант використання UC001

Назва	Реєстрація користувача.
Опис	Користувач має змогу зареєструватися.
Учасники	Користувач.
Передумови	—
Післяумова	Користувач успішно зареєстрований до системи.
Основний сценарій	1. Користувач переходить до сторінки реєстрації у web-інтерфейсі; 2. На сторінці присутня форма з полями вводу – електронна пошта, пароль, підтвердження паролю і неактивною кнопкою «авторизувати»; 3. Користувач заповнює поля вводу свої даними; 4. Кнопка «авторизувати» стає активною; 5. Користувач натискає кнопку «авторизувати»; 6. Система зареєструвала користувача, та перенаправила його до сторінки з вибором типу профіля.

Продовження таблиці 1.2 – Варіант використання UC001

Розширений сценарій	5.1 Система виявляє, що введені данні введені некоректно; 5.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.3 - Варіант використання UC002

Назва	Вибір типу профіля учень/студент.
Опис	Користувач обирає тип профіля учень/студент.
Учасники	Користувач.
Передумови	Користувач має буди щойно зареєстрованим та авторизованим.
Післяумова	Користувача направлено до сторінки створення профілю учня/студента.
Основний сценарій	1. Користувач переходить до сторінки вибору типу профіля у web-інтерфейсу; 2. На сторінці присутні 3 картки-посилання 3. (учень, викладач, представник навчального закладу); 4. Користувач натискає на картку учня; 5. Система направляє користувача до сторінки створення профіля учня.
Розширений сценарій	—

Таблиця 1.4 - Варіант використання UC003

Назва	Вибір типу профіля вчитель/викладач.
Опис	Користувач обирає тип профіля вчитель/викладач.
Учасники	Користувач.

Продовження табиці 1.4 – Варіант використання UC003

Передумови	Користувач має буди щойно зареєстрованим та авторизованим.
Післяумови	Користувача направляє на сторінку створення профіля вчителя/викладача.
Основний сценарій	1. Користувач переходить до сторінки вибору типу профіля у web-інтерфейсу; 2. На сторінці присутні 3 картки-посилання (учень, викладач, представник навчального закладу); 3. Користувач натискає на картку вчителя/викладача; 4. Система направляє користувача до сторінки створення профіля вчителя/викладача.
Розширений сценарій	—

Таблиця 1.5 - Варіант використання UC004

Назва	Вибір типу профіля представник навчального закладу.
Опис	Користувач обирає тип профіля представник навчального закладу.
Учасники	Користувач.
Передумови	Користувач має буди щойно зареєстрованим та авторизованим.
Післяумова	Користувача направлено до сторінки створення профілю представника навчального закладу.
Основний сценарій	1. Користувач переходить до сторінки вибору типу профіля у web-інтерфейсу;

Продовження таблиці 1.5 – Варіант використання UC004

Основний сценарій	2. На сторінці присутні 3 картки-посилання (учень, викладач, представник навчального закладу); 3. Користувач натискає на картку представник навчального закладу; 4. Система направляє користувача до сторінки створення профіля представника навчального закладу.
Розширений сценарій	—

Таблиця 1.6 - Варіант використання UC005

Назва	Створення профілю учня/студента.
Опис	Користувач створює профіль учень/студент.
Учасники	Користувач.
Передумови	Користувач має буди авторизованим.
Післяумова	Користувача направлено до сторінки з доступними оінюваннями (тестами).
Основний сценарій	1. Користувач переходить до сторінки створення профіля учня/студента у web-інтерфейсі; 2. На сторінці присутні поля вводу які необхідно заповнити – ПІБ, навчальний заклад (факультет, кафедра, якщо це університет/інститут), курс та номер групи (класу, якщо школа) та кнопка «зберегти».

Продовження таблиці 1.6 - Варіант використання UC005

Основний сценарій	3. Користувач натискає на кнопку зберегти; 4. Система направляє користувача до сторінки з доступними оцінюваннями.
Розширений сценарій	3.1 Система виявляє некоректно введені дані; 3.2 Система сповіщає користувача про виниклу помилку.

Таблиця 1.7 - Варіант використання UC006

Назва	Створення профілю вчителя/викладача
Опис	Користувач створює профіль вчителя/викладача.
Учасники	Користувач.
Передумови	Користувач має буди авторизованим.
Післяумова	Користувача направлено до сторінки з уже створеними оцінюваннями (тестами).
Основний сценарій	Користувач переходить до сторінки створення профіля у web-інтерфейсу; На сторінці присутні поля вводу які необхідно заповнити – ПІБ, навчальний заклад (факультет, кафедра, якщо це університет/інститут), та кнопка «зберегти». Користувач натискає на кнопку зберегти; Система направляє користувача до сторінки з уже створеними оцінюваннями (тестами).
Розширений сценарій	3.1 Система виявляє некоректно введені дані; 3.2 Система сповіщає користувача про виниклу помилку.

Таблиця 1.8 - Варіант використання UC007

Назва	Створення профілю представника навчального закладу.
Опис	Користувач створює профіль представника навчального закладу.
Учасники	Користувач.
Передумови	Користувач має бути авторизованим.
Післяумова	Користувача направлено до сторінки створення заявки на реєстрацію навчального закладу.
Основний сценарій	1. Користувач переходить до сторінки створення профіля у web-інтерфейсу; 2. На сторінці присутні поля вводу які необхідно заповнити – ПІБ, навчальний заклад та кнопка «зберегти». 3. Користувач натискає на кнопку зберегти; 4. Система направляє користувача до сторінки заявки на реєстрацію навчального закладу.
Розширений сценарій	3.1 Система виявляє некоректно введені дані; 3.2 Система сповіщає користувача про виниклу помилку.

Таблиця 1.9 – Варіант використання UC008

Назва	Авторизація адміністратора системи.
Опис	Адміністратор має змогу авторизації.
Учасники	Адміністратор.

Продовження таблиці 1.9 – Варіант використання UC008

Передумови	Адміністратор має бути зареєстрованим у системі.
Післяумова	Адміністратор успішно авторизований до системи.
Основний сценарій	1. Адміністратор переходить до сторінки авторизації у web-інтерфейсі; 2. На сторінці присутня форма з полями вводу – електронна пошта і пароль і неактивною кнопкою «авторизувати»; 3. Адміністратор заповнює поля вводу свої даними; 4. Кнопка «авторизувати» стає активною; 5. Адміністратор натискає кнопку «авторизувати»; 6. Система авторизувала адміністратора, та перенаправила його до адміністративної панелі.
Розширений сценарій	5.1 Система виявляє, що введені данні введені некоректно; 5.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.10 - Варіант використання UC009

Назва	Авторизація користувача.
Опис	Користувач авторизується в системі.
Учасники	Неавторизований користувач.
Передумови	Користувач має буди щойно зареєстрованим та авторизованим.
Післяумова	Користувача направлено до сторінки визначеної системою за замовчуванням.

Продовження таблиці 1.10 – Варіант використання UC009

Основний сценарій	1. Користувач переходить до сторінки авторизації у web-інтерфейсі; 2. На сторінці присутня форма з полями вводу – електронна пошта і пароль і неактивною кнопкою «авторизувати»; 3. Користувач заповнює поля вводу свої даними; 4. Кнопка «авторизувати» стає активною; 5. Користувач натискає кнопку «авторизувати»; 6. Система авторизувала Користувач, та перенаправила його до сторінки визначеної системою за замовчуванням.
Розширений сценарій	5.1 Система виявляє, що введені данні введені некоректно; 5.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.11 - Варіант використання UC010

Назва	Створення заявки про реєстрацію навчального закладу.
Опис	Користувач-представник навчального закладу створює заявку про реєстрацію навчального закладу.
Учасники	Користувач-представник навчального закладу.
Передумови	Користувач-представник має бути авторизований.
Післяумова	Користувач успішно створив заявку на реєстрацію навчального закладу.
Основний сценарій	1. Користувач-представник навчального закладу переходить до сторінки створення заявки на реєстрацію навчального заклад у web-інтерфейсі;

Продовження таблиці 1.11 - Варіант використання UC010

Основний сценарій	2. На сторінці присутня форма з полями вводу – назва навчального закладу, документ-посвідчення про акредитацію навчального закладу та кнопка «надіслати»; 3. Користувач-представник навчального закладу заповнює поля вводу; 4. Користувач натискає кнопку «надіслати»; 5. Система успішно створила заявку на реєстрацію навчального закладу.
Розширений сценарій	4.1 Система виявляє, що введені данні введені некоректно; 4.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.12 – Варіант використання UC011

Назва	Проходження оцінювання (тесту) учнем/студентом.
Опис	Користувач-учень/студент проходить оцінювання (тест).
Учасники	Користувач-учень/студент.
Передумови	Користувач має бути авторизованим та мати доступ до проходження оцінювання (тесту).
Післяумова	Користувач пройшов оцінювання та його було переадресовано на сторінку з результатом оцінювання (тесту).
Основний сценарій	1. Користувач переходить до сторінки з доступними оцінюваннями (тестами) у web-інтерфейсі;

Продовження таблиці 1.12 – Варіант використання UC011

Основний сценарій	2. Користувач-учень/студент натискає кнопку пройти тест та переходить на сторінку з детальним описом оцінювання та кнопкою «розпочати»; 3. Користувач-учень/студент натискає кнопку «розпочати»; 4. Система переадресує користувача-учня/студента на сторінку з усіма питаннями, варіантами відповідей для кожного з них та кнопкою «завершити»; 5. Користувач-учень/студент обирає варіанти відповідей та натискає кнопку «завершити»; 6. Система підраховує результат оцінювання (тесту) та переадресує користувача-учня/студента на сторінку з результатами оцінювання.
Розширений сценарій	—

Таблиця 1.13 – Варіант використання UC012

Назва	Створення оцінювання (тесту).
Опис	Користувач-вчитель/викладач створює оцінювання.
Учасники	Користувач-вчитель/викладач.
Передумови	Користувач-вчитель/викладач має бути авторизованим та підтвердженим користувачем-представником навчального закладу, до якого він належить.
Післяумова	Користувач-вчитель/викладач успішно створив оцінювання та його переадресовано на сторінку з детальною інформацією про оцінювання.

Продовження таблиці 1.13 – Варіант використання UC012

Основний сценарій	1. Користувач-вчитель/викладач переходить до сторінки створення оцінювання у web-інтерфейсі; 2. На сторінці присутня форма з полями вводу – назва оцінювання, форма додавання питань та варіантів відповідей та кнопка «зберегти»; 3. Користувач-вчитель/викладач заповнює поля вводу та наповнює оцінювання питаннями за допомогою форми; 4. Користувач натискає кнопку «зберегти»; 5. Система успішно створила оцінювання та перенаправила користувача-вчителя/викладача до сторінки з переліком питань до оцінювання
Розширений сценарій	4.1 Система виявляє, що данні було введено некоректно; 4.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.14 – Варіант використання UC013

Назва	Перегляд статистики.
Опис	Користувач-вчитель/викладач має змогу переглядати статистику здач на власній кафедрі (школі).
Учасники	Користувач-вчитель/викладач.
Передумови	Користувач-вчитель/викладач має бути авторизованим та підтвердженим користувачем-представником навчального закладу, до якого він належить.
Післяумова	—

Продовження таблиці 1.14 – Варіант використання UC013

Основний сценарій	1. Користувач переходить до сторінки зі статистикою у web-інтерфейсі; 2. На сторінці присутня форма з полями вводу – оцінювання, група та кнопка «відобразити»; 3. Користувач обирає дані у полях вводу; 4. Користувач натискає кнопку «відобразити»; 5. Система генерує статистичні дані здач та відображає їх на сторінці.
Розширений сценарій	4.1 Система виявляє, що введені данні введені некоректно або поля були залишені порожніми; 4.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.15 – Варіант використання UC014

Назва	Огляд власних результатів.
Опис	Користувач-учень/студент має змогу переглядати свої власні результати пройдених оцінювань.
Учасники	Користувач-учень/студент.
Передумови	Користувач-учень/студент має бути авторизований та мусить мати хоча б одне пройдене оцінювання.
Післяумова	—
Основний сценарій	1. Користувач переходить до сторінки з результатами у web-інтерфейсі; 2. Система отримала дані з результатами користувача, та відобразила їх на сторінці.
Розширений сценарій	1. Система виявляє, що користувач не має результатів та відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.16 – Варіант використання UC015

Назва	Редагування профілю користувача.
Опис	Користувач має змогу редагувати власний профіль.
Учасники	Користувач.
Передумови	Користувач має буди авторизований.
Післяумова	Користувач успішно відредагував власний профіль.
Основний сценарій	1. Користувач переходить до сторінки свого профілю у web-інтерфейсі; 2. На сторінці присутня інформація профілю та кнопка «редагувати»; 3. Користувач натискає кнопку «редагувати»; 4. На сторінці з'являється форма з полями вводу для редагування (всі поля профілю, крім парою та пошти) та кнопка «зберегти»; 5. Користувач натискає кнопку «зберегти»; 6. Система застосувала зміни та повернула користувача до попередньої сторінки.
Розширений сценарій	5.1 Система виявляє, що введені данні введені некоректно; 5.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.17 – Варіант використання UC016

Назва	Редагування профілю навчального закладу.
Опис	Користувач має змогу зареєструватися.
Учасники	Користувач-представник навчального закладу.

Продовження таблиці 1.17 – Варіант використання UC016

Передумови	Користувач-представник має бути авторизований та навчальний заклад має бути зареєстрованим та підтвердженим адміністратором системи.
Післяумова	Зміни успішно збережено. Система переадресує користувача-представника на сторінку з інформацією про навчальний заклад.
Основний сценарій	1. Користувач-представник навчального закладу переходить до сторінки профілю навчального закладу у web-інтерфейсі; 2. На сторінці присутня інформація про навчальний заклад та кнопка «редагувати»; 3. Користувач-представник навчального закладу натискає кнопку «редагувати»; 4. На сторінці з'являється форма з полями вводу для редагування та кнопка «зберегти»; 5. Користувач-представник навчального закладу натискає кнопку «зберегти»; 6. Система застосувала зміни та повернула користувача до попередньої сторінки.
Розширений сценарій	5.1 Система виявляє, що введені данні введені некоректно; 5.2 Система відображає повідомлення з інформацією про відповідну помилку.

Таблиця 1.18 – Варіант використання U017

Назва	Вихід із системи.
Опис	Користувач має змогу вийти із системи.
Учасники	Користувач.

Продовження таблиці 1.18 – Варіант використання U017

Передумови	Користувач має бути авторизованим.
Післяумова	Користувач успішно вийшов з системи.
Основний сценарій	1. Користувач переходить до сторінки виходу із системи у web-інтерфейсі; 2. На сторінці присутня кнопка «залишити систему»;
Основний сценарій	3. Користувач натискає кнопку «залишити систему»; 4. Система зберігає дані сесії користувача, виконує вихід його із системи та переадресує його на домашню сторінку web-інтерфейсу.
Розширений сценарій	—

Функціональні вимоги системи відповідають представленим вище варіантам використання та повинні бути реалізовані в повному обсязі. Також, перелічені вище варіанти використання зображені на діаграмі варіантів використання у додатку 1.

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинно мати наступний ряд нефункціональних вимог:

- кросплатформеність – доступ до системи повинен бути реалізований для будь-яких платформ за допомогою веб-браузера (персональні, комп’ютери, ноутбуки, мобільні пристрої);
- клієнт-серверна архітектура повинна буди реалізована у вигляді SSR (Server-Side Rendering) з використанням шаблону проектування MVC (Model View Controller);
- взаємодія з системою має проводитися лише з використанням захищеного SSL-з’єднання з використанням протоколу HTTPS;

– локалізація web-інтерфейсу та панелі адміністратора повинна бути виконана українською мовою.

1.5 Висновки до розділу

Проаналізувавши основні функціональні та нефункціональні вимоги до системи, а також вивчивши ринок готових рішень був сформований ряд задач, які система повинна виконувати для досягнення основної мети - розв’язання проблем несправедливої оцінки здобутих учнями/студентами теоретичних та/або практичних знань:

– Створення вчителями/викладачами навчальних закладів оцінювань (тестів) для оцінки рівня обізнаноті учня/студента у певній темі;
– Можливість для учнів/студентів проходити доступні для них тести, які в свою чергу будуть справедливо оцінені системою;
– Можливість навчальним закладам пройти реєстрацію задля підтвердження акредитованості навчального закладу;
– Реєстрація та авторизація користувача;
– Створення профіля визначеного типу для розгалуження функціоналу та доступу для нього;
– Демонстрація статистики здач для вчителів/викладачів та перегляд учнями/студентами власних результатів.

Система не повинна вимагати від користувача додаткових налаштувань. Вона повинна бути автоматизованою з середини та просити від користувача лише мінімальний об’єм даних для ідентифікації користувача.

2 ВИБІР ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

Наразі, існує безліч технологій розробки за допомогою яких можливо реалізувати будь-яку систему на архітектурі клієнт-сервер. Серед таких, мови програмування Python [1], Ruby, PHP, C#, Java та їх відповідні фреймворки Django (Python)[2], Ruby on Rails (Ruby), Laravel/Symphony/Yii (PHP), ASP.NET (C#), Spring (Java).

Проаналізувавши аналіз вимог до програмного забезпечення, для розробки даної системи була обрана мова програмування Python та її фреймворк Django разом із СКБД PostgreSQL [3] – для побудови серверної частини та HTML [4], CSS [5], JavaScript [7] та Bootstrap 4 [6] – для побудови web-інтерфейсу.

Фреймворк – це певна надбудова (каркас) над базовим функціоналом мови програмування, що створений задля полегшення побудови комплексних та складних систем.

СКБД – система керування базами даних. Є набором із самої бази даних та інструментів для створення, збереження і відтворення інформації.

2.1 Опис технологій розробки серверної частини

Основним інструментом у побудуванні будь-якого програмного забезпечення – є мова програмування. Для побудування системи була обрана мова програмування Python версії 3.7 [11].

Python – високорівнева мова програмування. Є інтерпретованою та має типізацію динамічного типу Вона є легкою в освоєнні та має дуже легкий і читабельний синтаксис. Має вбудовані можливості структур даних високого рівня з динамічною семантикою та динамічним зв’язуванням – це робить мову привабливою для швидкої розробки програмного забезпечення

Синтаксис мови програмування – це набір правил та мовних конструкцій (певна комбінація символів алфавіту) за допомогою яких описуються інструкції у виконуваному коді.

Семантика мови програмування – це смислове значення ключових слів та мовних конструкцій, операторів, тощо.

Далі наведено основні переваги і недоліки обраної мови програмування Python для побудови обраної системи.

Основні переваги:

- прозорий та легкий для сприйняття синтаксис. При розділення блоків виконуваного коду використовуються відступи;
- вбудовані бібліотеки і модулі мають безліч корисних та потужних інструментів;
- структури даних високого рівня мають велику ефективність;
- об’єктно-орієнтоване програмування має простий, але дуже ефективний підхід;
- дуже зручний інструмент для розв’язання математичних та статистичних задач.

Основні недоліки:

- низька швидкодія, як і у багатьох інтерпретованих мов. Але, це компенсується високою швидкістю розробки та за допомогою вбудованих в мову компонентів;
- відсутня можливості змінювати вбудовані класи, але це дозволяє Python використовувати менше набагато менший об’єм оперативної пам’яті, що надає змогу підвищити швидкість виконання програм;
- глобальне блокування інтерпретатора – в один момент часу може виконуватись лише одна нить (thread) коду Python. Але цей недолік можна компенсувати, використовуючи потужні засоби асинхронного виконання коду.

Разом з обраною мовою програмування був обраний допоміжний інструмент, а саме фреймворк Django версії 2.2 [12].

Django [13] – фреймоврк з відкритим кодом для мови програмування Python, основне призначення якого – розробка web-систем. При побудові системи за допомогою даного фреймворку, система поділяється на так звані застосунки (applications) – модулі з яких складається повноцінна система. Це є однією з особистостей даного фреймворку. Архітектура фреймворку базується на шаблоні проектування Model View Controller (MVC), але має деякі особливості: те що зазвичай називається «контролером» (controller), у Django називається «вид» (view), а «вид» - «шаблоном» (template). Тому архітектуру застосунків, побудованих за допомогою даного фреймворку називають Model Template View (MTV).

Основними можливостями фреймворку є:

- Object-Relational Mapping (ORM) – прикладний інтерфейс доступу до бази даних, який підтримує транзакції;
- будь-який проект, який обрав Django матиме інтерфейс (панель) адміністратора «із коробки»;
- вбудовані та легкі у налаштуванні системи кешування;
- вбудована система шаблонів, яка полегшує роботу з зовнішною частиною програми. Система шаблонів містить власний набір тегів (формалізованих виразів) та підтримує наслідування;
- інтернаціоналізація;
- можливість побудови так званих «форм», які унаслідуються від існуючої в базі даних моделі та полегшують валідацію;
- зручна побудова URL, як за допомогою регулярних виразів так і за допомогою визначеного синтаксису;
- вбудований web-сервер, який призначений для використання під час розробки. При змінах у файлах програми, сервер перезавантажується та застосовує зміни.

Разом з мовою програмування Python та її фреймворком Django, для побудови серверної частини системи використовується система керування базами даних PostgreSQL.

PostgreSQL – система керування базами даних. Є об’єктно-реляційною СКБД з відкритим вихідним кодом. Далі, наведено основні функціональні можливості даної СКБД:

- підтримка транзакцій;
- типи даних, які може задавати користувач;
- наслідування;
- присутня перевірка сумісності версій;
- зовнішні ключі (цілісність посилань);

Основні переваги PostgreSQL:

- PostgreSQL - безкоштовне ПЗ з відкритим вихідним кодом;
- цілісність даних - коли надійність і цілісність даних є основними вимогами, PostgreSQL буде, найкращим вибором;
- підтримка спільноти. PostgreSQL має досить велику спільноту, яка допоможе знайти відповіді на поставлені запитання;
- доповнення – PostgreSQL має безліч вбудованих функцій, однак спільнота створила чималу кількість доповнень, що дозволяють розробляти дані для цієї СКБД та керувати ними;
- існує можливість зберігати власні процедури, задля розширення функціоналу;
- підтримка наслідування та об’єктність є перевагами PostgreSQL, як об’єктно-реляційною СКБД.

Основні недоліки системи керування базами даних PostgreSQL:

- невисока продуктивність – проводячи прості операції читання дана СКБД може витрати більше часу, аніж конкуренти;

– низька популярність – хоча PostgreSQL має свою спільноту – вона у порівняні менша ніжу конкурентів;

Основним фактором при виборі СКБД для серверної частини системи є саме надійність а цілісність даних. Саме тому було обрано саме PostgreSQL.

2.2 Опис технологій розробки web-інтерфейсу

Для розробки web-інтерфейсу системи, тобто зовнішньої її частини, було використано наступні технології: HTML, CSS, JavaScript, Bootstrap та мова шаблонів Django.

Основним інструментом розробки будь-якого web-інтерфейсу є мова розмітки гіпертексту – HTML. HTML – є так званим будівником web-сторінки. За допомогою визначеного синтаксису (тегів) розмічаються елементи сторінки і за допомогою каскадної таблиці стилів - CSS до цих елементів застосовуються стилі. [14]

Мова програмування JavaScript у даному контексті виступає як інструмент додання певної плавності, гнучкості і краси до web-сторінок. [15]
За допомоги цієї мови програмування можна «оживити» сторінку – додати анімацій, плавних переходів тощо.

Bootstrap – це набір інструментів і шаблонів для побудови web-інтерфейсів. Він значно полегшує задачу розробки завдяки присутності готових рішень. Достатньо лише додати до елемента сторінки у HTML-коді певний клас, і Bootstrap автоматично застосує до цього елемента певні стилі, чи функціонал.

Мова шаблонів Django – це набір формалізованих виразів та тегів, визначених власним синтаксисом, що дозволяють полегшити розробку зовнішньої частини системи за допомогу наслідувань, розширень, додавання зовнішніх HTML-сторінок, тощо.

2.3 Висновки до розділу

Проаналізувавши аналіз вимог до програмного забезпечення, для розробки даної системи була обрана мова програмування Python та її фреймворк Django разом із СКБД PostgreSQL – для побудови серверної частини та HTML, CSS, JavaScript та Bootstrap 4 – для побудови web-інтерфейсу.

За допомогою даних технологій побудувати потрібну систему буде легко, швидко та не треба буде докладати додаткових зусиль. Використовуючи сучасні технології розробки web-інтерфейсів (HTML, CSS, JavaScript, Bootstrap і мова шаблонів Django) та потужні можливості мови програмування Python та її фреймворку Django (для побудови серверної частини) – розробка даної системи не потребуватиме великої кількості часу та зусиль. Дані технології є передовими у власних сферах, та вдосконалюються кожен день для покращення та полегшення циклу розробки програмного забезпечення

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В даному розділі представлена реалізація програмного забезпечення. Так як вибрана технологія розробки, а саме фреймворк Django, використовує модель MTV (Model Template View), то web-інтерфейс та серверна частина дуже пов’язані між собою, на рівні реалізації. Однак, їх необхідно розглядати окремо, для кращого розуміння реалізації програмного продукту. Першим розглянуто web-інтерфейс.

3.1 Реалізація web-інтерфейсу (Template)

Так, як програмне забезпечення реалізовано за допомогою шаблона проектування Model View Controller (MVC), а саме, за допомогою його різновиду Model Template View, то сам web-інтерфейс реалізовано у шаблонах (Template) та система і користувач доступаються до них за допомогою виду (View). Далі зроблений опис основних мовних конструкцій мови шаблонів Django, які були використані разом з мовою HTML для побудови web-інтерфейсу:

- {% load static %} – дана конструкція повідомляє системі, що для даного шаблону необхідно завантажити теку під назвою «static». В даній теці зберігаються так звані «статичні файли» - зображення, каскадні таблиці стилів CSS, файли коду JavaScript, шрифти, тощо. Все, що необхідно для статичного відображення сторінки;
- {% block title %}{% endblock %} – спеціальна мовна конструкція мови шаблонів Django. Дана конструкція резервує місце в шаблоні під блок з назвою «title», яке в подальшому буде замінене на код з іншого шаблону. Дана конструкція дозволяє уникнути повторного використання блоків коду в різних HTML файлах та дозволяє динамічно завантажувати вміст однієї сторінки в іншу, без додаткових зусиль;
- {% static 'img/logo.png' %} – дана конструкція мови шаблонів Django завантажує релевантний шлях до «статичного» файлу, що

знаходиться у теці «static» та замінює себе на даний шлях. В основному, використовується для завантаження зображень, шрифтів, стилів, JavaScript скриптів до HTML сторінки;

- {% if user.is_authenticated %}{% endif %} – мовна конструкція умови. В даному випадку перевіряє, чи користувач авторизований до системи;
- {% url 'index_app:signout' %} – мовна конструкція мови шаблонів Django, що завантажує уніфікований локатор ресурсів до певного виду. Конструкція «index_app:signout», повідомляє системі, що потрібно завантажити вид модуля «signup», який знаходиться у застосунку (application) «index_app».
- конструкція {% for x in list%}{% endfor %} – дає змогу виконати циклічні ді – наприклад, пройти циклом по набору елементів, та вивести один з атрибутів до HTML сторінки. Дана конструкція не зазначена у прикладі;
- конструкція {{ user.email }}– це конструкція змінної. Дані що містяться у змінній «user.email» підставляються замість даної мовної конструкції.

Дані мовні конструкції дозволяють швидко та динамічно, без зайвих зусиль наповнити HTML сторінку вмістом.

3.2 Реалізація контролерів (View)

У даному підрозділі розглянуто ключові методи контролерів системи (View у Model Template View (MTV)), які приймають участь у побудові та відображені web-інтерфейсу а також у повному функціонуванні системи.

Нижче у таблиці 3.1 «Опис методів авторизації та реєстрації» наведено опис методів реєстрації та авторизації користувачів.

Таблиця 3.1 – Опис методів авторизації та реєстрації

Назва методу	Опис	Використані вбудовані бібліотеки та модулі
signup(request)	Метод реєстрації користувачів. На вхід до методу, подається аргумент «request» - що є надісланим запитом до системи. При вдалому сценарії- повертає HTTP-відповідь з переадресацією на сторінку вибору типу профіля. При невдалому – повертає HTTP-відповідь з переадресацією на сторінку з реєстрацію, а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання.	django.shortcuts.render, django.shortcuts.redirect, django.contrib.auth.login, django.contrib.auth.authenticate, шаблон«index_app/signup.html»
signin(request)	Метод авторизації користувачів. На вхід до методу подається аргумент «request» - що є надісланим запитом до системи. З POST-запиту отримує параметри email та password та за допомогою їх авторизує користувача у системі. Якщо, користувача успішно	django.shortcuts.render, django.shortcuts.redirect, django.contrib.auth.login, django.contrib.auth.authenticate, шаблон «index_app/signin.html»

Продовження таблиці 3.1 – Опис методів авторизації та реєстрації

Назва методу	Опис	Використані вбудовані бібліотеки та модулі
signin(request)	авторизовано до системи – повертає HTTP-відповідь з переадресацією до сторінки передбаченою типом профіля користувача (учень/студент, вчитель/викладач, представник навчального закладу або адміністратор). При невдалому – повертає HTTP-відповідь з переадресацією на сторінку з авторизацією, а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання.	
signout(request)	Метод виходу користувача з системи. На вхід до методу подається аргумент «request» - що є надісланим запитом до системи. Оброблює вихід користувача з системи. Якщо був надісланий POST-запит виконує вихід користувача з системи та повертає HTTP-	django.shortcuts.render, django.shortcuts.redirect, django.contrib.auth.logout, шаблон«index_app/signout.html»

Продовження таблиці 3.1 – Опис методів авторизації та реєстрації

Назва методу	Опис	Використані вбудовані бібліотек модулі
signout(request)	відповідь з переадресацією на сторінку з авторизацією. В іншому випадку – якщо був виконаний GET-запит – повертає HTTP-відповідь з переадресацією на сторінку з виходу з системи.	

Нижче у таблиці 3.2 «Опис методів створення профілю» наведено опис методів для створення профілю трьох типів: учень/студент, вчитель/викладач або представник навчального закладу.

Таблиця 3.2 – Опис методів створення профілю

Назва методу	Опис
profile_select(request)	Метод вибори типу профіля. На вхід до методу, подається аргумент «request» - що є надісланим запитом до системи. При вдалому сценарії, якщо буд надісланий POST-запит разом з вибором користувача - повертає HTTP-відповідь з переадресацією на сторінку створення профіля обраного типу. При GET-запиті – повертає HTTP-відповідь з переадресацією на сторінку з вибором типу профілю. Використанні вбудовані модулі та бібліотеки:

Продовження таблиці 3.2 – Опис методів створення профілю

Назва методу	Опис
profile_select(request)	– django.shortcuts.render; – django.shortcuts.redirect.
profile_create_student(request)	Метод створення профілю типу учень/студент. На вхід до методу подається аргумент «request» - що є надісланим запитом до системи. З POST-запиту отримує параметри з форми, яка знаходиться у HTML файлі та перевіряє їх на правильність за допомогою так званих «форм» - класів які містять вбудовані методи перевірки відповідності отриманих даних. Якщо, користувач успішно створив профіль – метод повертає HTTP-відповідь з переадресацією до сторінки з доступними для учня/студента оцінюваннями (тестами). При невдалому створення профіля– повертає HTTP-відповідь з переадресацією на сторінку створення профіля учня/тудента, а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання. Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; – django.shortcuts.redirect.
profile_create_teacher(request)	Метод створення профілю типу вчитель/викладач. На вхід до методу подається аргумент «request» - що є надісланим запитом до системи. З POST-запиту отримує параметри

Продовження таблиці 3.2 – Опис методів створення профілю

Назва методу	Опис
profile_create_teacher(request)	з форми, яка знаходиться у HTML файлі та перевіряє їх на правильність за допомогою так званих «форм» - класів які містять вбудовані методи перевірки відповідності отриманих даних. Якщо, користувач успішно створив профіль – метод повертає HTTP-відповідь з переадресацією до сторінки зі створеними оцінюваннями (тестами) та статистикою. При невдалому створення профіля– повертає HTTP-відповідь з переадресацією на сторінку створення профіля вчитея/викладача, а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання. Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; – django.shortcuts.redirect.
profile_create_repr(request)	Метод створення профілю типу представника навчального закладу. На вхід до методу подається аргумент «request» - що є надісланим запитом до системи. З POST-запиту отримує параметри з форми, яка знаходиться у HTML файлі та перевіряє їх на правильність за допомогою так званих «форм» - класів які містять вбудовані методи перевірки відповідності отриманих даних. Якщо, користувач успішно створив профіль – метод

Продовження таблиці 3.2 – Опис методів створення профілю

Назва методу	Опис
profile_create_repr(request)	повертає HTTP-відповідь з переадресацією до сторінки зі створенням заявки на реєстрацію навчального закладу. При невдалому створення профіля– повертає HTTP-відповідь з переадресацією на сторінку створення профіля представника навчального закладу, а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання. Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; django.shortcuts.redirect.

Нижче у таблиці 3.3 «Опис методів створення оцінювання (тесту)» наведено опис методів для створення вчителем/викладачем оцінювання (тесту).

Таблиця 3.3 – Опис методів створення оцінювання (тесту)

Назва методу	Опис
quizzes_create(request)	Метод створення оцінювання (тесту). На вхід до методу подається аргумент «request» - що є надісланим запитом до системи. З POST-запиту отримує параметри з форми, яка знаходиться у HTML файлі та перевіряє їх на правильність за допомогою так званих «форм» - класів які містять вбудовані методи перевірки відповідності отриманих даних. Якщо, користувач успішно створив профіль – метод повертає HTTP-відповідь з переадресацією до сторінки з

Продовження таблиці 3.3 – Опис методів створення оцінювання (тесту)

Назва методу	Опис
quizzes_create(request)	інформацією про оцінювання (тест). При невдалому створення профіля– повертає HTTP-відповідь з переадресацією на сторінку створення оцінювання (тесту), а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання. Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; – django.shortcuts.redirect; – django.contrib.auth.decorators.login_required.
quiz_add_questions(request, slug)	Метод додавання питання при створенні оцінювання (тесту). На вхід до методу подається аргумент «request» - що є надісланим запитом до системи та «slug» - унікальний ідентифікатор оцінвання. З POST-запиту отримує параметри з форми, яка знаходиться у HTML файлі та перевіряє їх на правильність за допомогою так званих «форм» - класів які містять вбудовані методи перевірки відповідності отриманих даних. Якщо, користувач успішно створив питання – метод зберігає питання та повертає HTTP-відповідь з переадресацією до сторінки з створенням оцінювання (тесту). При невдалому створенні питання – повертає HTTP-відповідь з переадресацією на сторінку створення оцінювання (тесту), а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання.

Продовження таблиці 3.3 – Опис методів створення оцінювання (тесту)

Назва методу	Опис
quiz_add_questions(request, slug)	Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; – django.shortcuts.redirect; – django.contrib.auth.decorators.login_required.
add_answer(request, slug)	Метод додавання варіанту відповіді до питання при створенні оцінювання (тесту). На вхід до методу подається аргумент «request» - що є надісланим запитом до системи та «slug» - унікальний ідентифікатор питання.. З POST-запиту отримує параметри з форми, яка знаходиться у HTML файлі та перевіряє їх на правильність за допомогою так званих «форм» - класів які містять вбудовані методи перевірки відповідності отриманих даних. Якщо, користувач успішно створив варіант відповіді – метод зберігає питання та повертає HTTP-відповідь з переадресацією до сторінки з створенням оцінювання (тесту). При невдалому створенні варіанта відповіді – повертає HTTP-відповідь з переадресацією на сторінку створення оцінювання (тесту), а в якості даних, які посилає разом HTTP-відповіддю – помилку та її описання. Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; – django.shortcuts.redirect;

Нижче у таблиці 3.4 «Опис методів проходження оцінювання (тесту)» наведено опис методів для проходження учнем/студентом оцінювання (тесту).

Таблиця 3.4 – Опис методів проходження оцінювання (тесту)

Назва методу	Опис
start_test(request, slug)	Метод який надає змогу учню/студенту розпочати оцінювання (тест). На вхід до методу подається аргумент «request» - що є надісланим запитом до системи та «slug» - унікальний ідентифікатор оцінювання (тесту). При GET-запиті метод повертає HTTP-відповідь з переадресацією до сторінки з розпочатим оцінюванням (тестом). Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; – django.shortcuts.redirect; – django.contrib.auth.decorators.login_required; – datetime.datetime; – datetime.timedelta.
stop_test(request, slug)	Метод який надає змогу учню/студенту завершити оцінювання (тест). На вхід до методу подається аргумент «request» - що є надісланим запитом до системи та «slug» - унікальний ідентифікатор оцінювання (тесту). При POST-запиті метод отримує відмічені учнем/студентом відповіді оброблює їх та повертає HTTP-відповідь з переадресацією до сторінки з результатом оцінюванням (тестом). Використанні вбудовані модулі та бібліотеки:

Продовження таблиці 3.4 – Опис методів проходження оцінювання (тесту)

Назва методу	Опис
stop_test(request, slug)	– django.shortcuts.render; – django.shortcuts.redirect; – django.contrib.auth.decorators.login_required
control_work_calculate_mark(answers)	Метод який підраховує відсоткову оцінку за пройдене оцінювання (тест). На вхід до методу подається аргумент «request» - що є надісланим запитом до системи та «answers» - відповіді які надав учень/студент. Використанні вбудовані модулі та бібліотеки: – math.
render_result(request, slug)	Метод який відображає учню/студенту результат за пройдене оцінювання (тест). На вхід до методу подається аргумент «request» - що є надісланим запитом до системи та «slug» - унікальний ідентифікатор оцінювання (тесту). При GET-запиті метод отримує результат учня/студента відповіді оброблює їх та повертає HTTP-відповідь з переадресацією до сторінки з результатом оцінюванням (тестом). Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render; – django.shortcuts.redirect; – django.contrib.auth.decorators.login_required.

Продовження таблиці 3.4 – Опис методів проходження оцінювання (тесту)

render_results(request)	Метод який надає змогу учню/студенту переглянути всі результати за пройдені оцінювання (тести) На вхід до методу подається аргумент «request» - що є надісланим запитом до системи. При GET-запиті метод повертає HTTP-відповідь з переадресацією до сторінки з усіма пройденими учнем/студентом оцінюваннями (тестами). Використанні вбудовані модулі та бібліотеки: – django.shortcuts.render.

3.3 Реалізація моделей (Model)

У даному підрозділі розглянуто ключові моделі системи (View у Model Template View (MTV)), які приймають участь у побудові та функціонуванні системи.

Нижче у таблиці 3.5 «Опис моделі Користувача» наведено опис моделі користувача системи. Для побудови моделі користувача системи була використана вбудована у фреймворк модель користувача (AbstractBaseUser) та було створено клас-управитель (UserManager), який керує створеннями об’єктів моделі користувача системи.

Нижче наведено перелік всіх базових моделей системи:

- користувач (User);
- профіль (Profile);
- навчальний заклад (EducationalInstitution);
- факультет (Faculty);
- кафедра (Department);

- оцінювання/тест (Quiz);
- питання (Question);
- варіант відповіді (Answer);
- результат (Result);

Таблиця 3.5 – Опис моделі Користувача

Назва моделі	Користувач (User)
Опис	Модель користувача системи.
Атрибути	– електрона пошта (email); – пароль (password); – дата та час останньої авторизації (last_login); – чи є даний користувач адміністратором (is_staff, is_superuser);
Методи	Користувач має всі вбудовані методи, які має клас AbstractBaseUser. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.6 «Опис моделі Профілю» наведено опис моделі профіля користувача системи.

Таблиця 3.6 – Опис моделі Профілю

Назва моделі	Профіль (Profile)
Опис	Модель профілю користувача системи.
Атрибути	– користувача (user) – зовнішній зв’язок (один до одного); – ім’я (first_name); – прізвище (last_name); – по батькові (middle_name); – кафедра (last_login); – навчальний заклад (educational_institution);

Продовження таблиці 3.6 - Опис моделі Профілю

Атрибути	– тип профілю (profile_type).
Методи	Профіль має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.7 «Опис моделі Навчального Закладу» наведено опис моделі Навчального Закладу.

Таблиця 3.7 – Опис моделі Навчального Закладу

Назва моделі	Навчальний заклад (EducationalInstitution)
Опис	Модель навчального закладу, що є однією з основних інстанцій системи.
Атрибути	– назва (title); – тип навчального закладу (ed_inst_type); – профіль представника навчального закладу (representative) – зовнішній зв’язок (один до одного).
Методи	Навчальний заклад має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.8 «Опис моделі Факультету» наведено опис моделі Факультету.

Таблиця 3.8 – Опис моделі Факультету

Назва моделі	Факультет (Faculty)
Опис	Модель факультету, що є однією з основних інстанцій системи.
Атрибути	– назва (title); – навчальний заклад (educational_institution) – зовнішній зв’язок (багато до одного).

Продовження таблиці 3.8 – Опис моделі Факультету

Методи	Факультет має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.9 «Опис моделі Кафедри» наведено опис моделі Кафедри.

Таблиця 3.9 – Опис моделі Кафедри

Назва моделі	Кафедра (Department)
Опис	Модель кафедри, що є однією з основних інстанцій системи.
Атрибути	– назва (title); – факультет (faculty) – зовнішній зв’язок (багато до одного).
Методи	Кафедра має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.10 «Опис моделі Оцінювання/Тесту» наведено опис моделі Кафедри.

Таблиця 3.10 – Опис моделі Оцінювання/Тесту

Назва моделі	Оцінювання/Тест (Quiz)
Опис	Модель Оцінювання/Тесту, що є однією з основних інстанцій системи.
Атрибути	– назва (title); – автор (author) – зовнішній зв’язок (багато до одного); – зображення (image); – короткий опис (short_description); – повний опис (full_description);

Продовження таблиці 3.10 – Опис моделі Оцінювання/Тесту

Атрибути	– дата та час створення (created); – час на складання (time).
Методи	get_questions(self) – метод, який повертає множину питань, які відносяться до даного оцінювання/тесту. Також модель Оцінювання/Тесту має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.11 «Опис моделі Питання» наведено опис моделі Питання.

Таблиця 3.11 – Опис моделі Питання

Назва моделі	Питання (Questions)
Опис	Модель Питання, що є однією з основних інстанцій системи.
Атрибути	– вміст питання (content); – оцінювання/тест (quiz) – зовнішній зв’язок (багато до одного); – кількість балів за питання (points); – декілька вірних відповідей (multiple_answers); – дата та час створення (created).
Методи	get_answers(self) – метод, який повертає множину відповідей, які відносяться до даного питання. Також модель Питання має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.12 «Опис моделі Варіанта Відповіді» наведено опис моделі Варіанта Відповіді.

Таблиця 3.12 – Опис моделі Варіанта Відповіді

Назва моделі	Варіант Відповіді (Answer)
Опис	Модель Варіант Відповіді, що є однією з основних інстанцій системи.
Атрибути	– вміст варіанту відповіді (content); – пиатння (question) – зовнішній зв’язок (багато до одного); – чи є варіант вірним (is_correct); – дата та час створення (created).
Методи	Модель Варіанту Відповіді має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

Нижче у таблиці 3.13 «Опис моделі Результату» наведено опис моделі Результату.

Таблиця 3.13 – Опис моделі Варіанта Відповіді

Назва моделі	Результат (Answer)
Опис	Модель Результату, що є однією з основних інстанцій системи.
Атрибути	– профіль учня/студента (profile) – зовнішній зв’язок (один до одного); – оцінювання/тест (quiz) – зовнішній зв’язок (багато до багатьох); – результати (results); – дата та час створення (created).
Методи	Модель Результату має всі вбудовані методи, які має клас django.db.models.Model. Всі методи описані на офіціальному сайті документації фреймворку Django [2].

3.4 Висновки до розділу

В даному розділі було розглянуто реалізацію програмного забезпечення, а саме основні компоненти системи – шаблони, контролери та моделі. Програмне забезпечення було побудовано на основі шаблону проектування Model-View-Controller, а саме за допомогою його різновиду у фреймворці Django – Model-Template-View.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення – одна із основних вимог до його розробки та повного функціонування без збоїв. Гарно протестоване програмне забезпечення є запорукою успіху запуску та життєдіяльності програмного забезпечення. Далі розглянуто основні методи тестування програмного забезпечення, які було використано при тестуванні побудованої системи.

4.1 Опис використаних методів тестування

Основним та обов’язковим тестуванням будь-якої системи – є модульне тестування. Модульне тестування - метод тестування програмного забезпечення, за допомогою якого перевіряються окремі одиниці вихідного коду, набори одного або більше комп'ютерних програмних модулів разом з відповідними контрольними даними, процедурами використання та операційними процедурами, щоб визначити, чи є вони придатними для використання.

Основними одиницями модельного тестування є сам модуль, який тестується та тестовий випадок – набір тестових інструкцій, за допомогою яких перевіряється обраний модуль.

Інший метод тестування, який було використано при тестуванні даного програмного забезпечення – це ручне тестування. Ручне тестування - процес ручного тестування програмного забезпечення на дефекти. Вона вимагає від тестера відігравати роль кінцевого користувача, завдяки чому вони використовують більшість функцій програми для забезпечення правильної поведінки. Щоб гарантувати повноту тестування, тестер часто дотримується письмового плану тестування, який веде їх через набір важливих тестових випадків.

4.2 Опис використаних технологій тестування

В даному підрозділі наведено опис технологій які було використано для модульного тестування програмного забезпечення (для ручного тестування необхідним є лише веб-браузер).

Основним інструментом модульного тестування для мови програмування Python є вбудований у мову тестувальний фреймворк «unittest».

Основними компонентами даного фреймворку є:

- «випробувальний прилад», що являє собою підготовку, необхідну для виконання одного або декількох тестів, а також будь-яких асоціативних дій очищення. Це може включати, наприклад, створення тимчасових або проксі-баз даних, каталогів або запуску серверного процесу;
- тестовий випадок, який є індивідуальною одиницею тестування. Він перевіряє конкретну відповідь на певний набір входів. unittest забезпечує базовий клас TestCase, який може бути використаний для створення нових тестів;
- набір тестів, що є набором тестових випадків, тестових наборів або обох. Він використовується для агрегації тестів, які повинні виконуватися разом;
- елемент запуску тестів - це компонент, який керує виконанням тестів і забезпечує результат користувачеві. Елемент запуску може використовувати графічний інтерфейс, текстовий інтерфейс або повертати спеціальне значення для позначення результатів виконання тестів.

4.3 Основні тестові випадки системи

В даному підрозділі у таблиці 4.1 «Тестові випадки розробленої системи» наведено всі тестові випадки системи

Таблиця 4.1 – Тестові випадки розробленої системи

ТС001	Користувач може авторизуватися до системи.
ТС002	Користувач може зареєструватися у системі.
ТС003	Користувач може створити профіль типу учень/студент.
ТС004	Користувач може створити профіль типу вчитель/викладач.
ТС005	Користувач може створити профіль типу представник навчального закладу.
ТС006	Вчитель/викладач може створити оцінювання (тест).
ТС007	Учень/студент може пройти оцінювання (тест).
ТС008	Представник навчального закладу може подати заявку на реєстрацію навчального закладу.
ТС009	Представник навчального закладу може редагувати профіль навчального закладу.
ТС010	Користувач може редагувати власний профіль.
ТС011	Вчитель/викладач може переглядати статистику.
ТС012	Учень/студент може переглядати результат.

Нижче, у підрозділі 4.4 «Опис тестових випадків системи» кожен приведений у таблиці 4.1 «Тестові випадки розробленої системи» вище тестовий випадок розглянуто окремо та більш детально.

4.4 Опис тестових випадків системи

В даному підрозділі кожен тестовий випадок розглянуто окремо і більш детально. Нижче у таблиці 4.2 «Опис тестового випадку ТС001» розглянуто тестовий випадок ТС001.

Таблиця 4.2 – Опис тестового випадку ТС001

Тестовий випадок	Користувач може авторизуватися до системи.
Передумови	Користувач повинен бути зареєстрованим.

Продовження таблиці 4.2 – Опис тестового випадку ТС001

Очікувана поведінка	– дані користувача введено вірно; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.3 «Опис тестового випадку ТС002» розглянуто тестовий випадок ТС002.

Таблиця 4.3 – Опис тестового випадку ТС002

Тестовий випадок	Користувач може зареєструватися у системі.
Передумови	—
Очікувана поведінка	– дані користувача введено вірно; – система зареєструвала користувача та зберегла його до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.4 «Опис тестового випадку ТС003» розглянуто тестовий випадок ТС003.

Таблиця 4.4 – Опис тестового випадку ТС003

Тестовий випадок	Користувач може створити профіль типу учень/студент.
Передумови	Користувач повинен бути зареєстрованим.
Очікувана поведінка	– дані користувача введено вірно; – система створила профіль типу учень/студент та зберегла його до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.5 «Опис тестового випадку ТС004» розглянуто тестовий випадок ТС004.

Таблиця 4.5 – Опис тестового випадку ТС004

Тестовий випадок	Користувач може створити профіль типу вчитель/викладач.
Передумови	Користувач повинен бути зареєстрованим.
Очікувана поведінка	– дані користувача введено вірно; – система створила профіль типу вчитель/викладач та зберегла його до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.6 «Опис тестового випадку ТС005» розглянуто тестовий випадок ТС005.

Таблиця 4.6 – Опис тестового випадку ТС005

Тестовий випадок	Користувач може створити профіль типу представник навчального закладу.
Передумови	Користувач повинен бути зареєстрованим.
Очікувана поведінка	– дані користувача введено вірно; – система створила профіль типу представник навчального закладу та зберегла його до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.7 «Опис тестового випадку ТС006» розглянуто тестовий випадок ТС006.

Таблиця 4.7 – Опис тестового випадку ТС006

Тестовий випадок	Вчитель/викладач може створити оцінювання (тест).
Передумови	Користувач повинен бути зареєстрованим та мати тип профілю вчитель/викладач.
Очікувана поведінка	– дані користувача введено вірно; – система створила оцінювання (тест) та зберегла його до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.8 «Опис тестового випадку ТС007» розглянуто тестовий випадок ТС007.

Таблиця 4.8 – Опис тестового випадку ТС007

Тестовий випадок	Учень/студент може пройти оцінювання (тест).
Передумови	Користувач повинен бути зареєстрованим та мати тип профілю учень/студент.
Очікувана поведінка	– дані користувача введено вірно; – система зберегла результати оцінювання (тесту) до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.9 «Опис тестового випадку ТС008» розглянуто тестовий випадок ТС008.

Таблиця 4.9 – Опис тестового випадку ТС008

Тестовий випадок	Представник навчального закладу може подати заявку на реєстрацію навчального закладу.

Продовження таблиці 4.9 – Опис тестового випадку ТС008

Передумови	Користувач повинен бути зареєстрованим та мати тип профілю представник навчального закладу.
Очікувана поведінка	– дані користувача введено вірно; – система створила заявку на реєстрацію навчального закладу та зберегла його до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.10 «Опис тестового випадку ТС009» розглянуто тестовий випадок ТС009.

Таблиця 4.10 – Опис тестового випадку ТС009

Тестовий випадок	Представник навчального закладу може редагувати профіль навчального закладу.
Передумови	Користувач повинен бути зареєстрованим та мати тип профілю представник навчального закладу.
Очікувана поведінка	– дані користувача введено вірно; – система змінила дані про навчальний заклад та зберегла їх до бази даних; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.11 «Опис тестового випадку ТС010» розглянуто тестовий випадок ТС010.

Таблиця 4.11 – Опис тестового випадку ТС010

Тестовий випадок	Користувач може редагувати власний профіль.
Передумови	Користувач повинен бути зареєстрованим.

Продовження таблиці 4.11 – Опис тестового випадку ТС010

Очікувана поведінка	– дані користувача введено вірно;
Очікувана поведінка	– система змінила дані про профіль користувача та зберегла їх до бази даних та повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.12 «Опис тестового випадку ТС011» розглянуто тестовий випадок ТС011.

Таблиця 4.12 – Опис тестового випадку ТС011

Тестовий випадок	Вчитель/викладач може переглядати статистику.
Передумови	Користувач повинен бути зареєстрованим та мати тип профілю вчитель/викладач.
Очікувана поведінка	– система надала дані про статистику здач; – система повернула код статусу 200.
Результат	Пройдено

Нижче у таблиці 4.13 «Опис тестового випадку ТС012» розглянуто тестовий випадок ТС012.

Таблиця 4.13 – Опис тестового випадку ТС012

Тестовий випадок	Учень/студент може переглядати результат.
Передумови	Користувач повинен бути зареєстрованим та мати тип профілю учень/студент.
Очікувана поведінка	– система надала дані про результати здач студента; – система повернула код статусу 200.
Результат	Пройдено

4.1 Висновки до розділу

Тестування програмного забезпечення – одна із основних вимог до його розробки та повного функціонування без збоїв. Гарно протестоване програмне забезпечення є запорукою успіху запуску та життєдіяльності програмного забезпечення. В даному розділі було розглянуто основні методи тестування програмного забезпечення, які було використано при тестуванні побудованої системи. А саме – модульне тестування та ручне тестування. Модульне тестування - метод тестування програмного забезпечення, за допомогою якого перевіряються окремі одиниці вихідного коду, набори одного або більше комп'ютерних програмних модулів разом з відповідними контрольними даними, процедурами використання та операційними процедурами, щоб визначити, чи є вони придатними для використання. Ручне тестування - процес ручного тестування програмного забезпечення на дефекти. Також, було розглянуто основні тестові випадки системи, які були створені на основі складених варіантів використання системи.

5 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В даному розділі розглянуто шлях та засоби за допомогою яких дане програмне забезпечення (систему) можна впроваджувати та запустити.

Для впровадження даної системи необхідно мати фізичний або віртуальний сервер (VPS) з такими мінімальними характеристиками:

- двоядерний процесор з тактовою частотою 2.4 ГГц;
- оперативна пам’ять об’ємом від 2 ГБ;
- жорсткий або твердотілий накопичувач об’ємом від 25ГБ;
- операційна система Ubuntu Server версії 16.04 (або інша Linux система призначена для серверів).

Для прикладу впровадження даної системи було обрано віртуальний приватний сервер (VPS) з операційною системою Ubuntu Server версії 16.04 з характеристиками наведеними вище. Постачальник послуги віртуального приватного серверу стала компанія ovh [].

5.1 Опис технологій впровадження

Для впровадження даного програмного забезпечення (системи) було використано технології наведені нижче:

- інструмент для керування розділеними Linux-контейнерами Docker;
- проксі-сервер Nginx;
- веб-сервер Gunicorn.

5.2 Docker

Як наведено вище, Docker – це інструмент для керування розділеними Linux-контейнерами. Він необхідний для того щоб зробити будь-яке програмне забезпечення легким у впровадженні на будь якій системі незалежно від того чи це робоча машина розробника даної системи, чи сервер на якому дане програмне забезпечення буде розгорнуто.

Ключовими одиницями Docker є образ і контейнер. Образ містить операційну систему, застосунок і всі його залежності. Образи в Docker складаються з шарів. Якщо нам треба образ з веб-сервером, то ми беремо за основу образ з дистрибутивом операційної системи, додаємо залежність - веб-сервер, і записуємо це як новий образ, який матиме два шари - один з ОС, наступний з веб-сервером. Контейнер Docker - це запущений образ. Контейнери Docker можна запускати, спиняти, переміщувати і видаляти.

5.3 Nginx

Nginx – це web-сервер та проксі сервер. Нижче наведено основні функції Nginx.

Основні функції Nginx, як HTTP-серверу:

- підтримка SSL;
- акселерована підтримка FastCGI і memcached серверів, простий розподіл навантаження і відмовостійкість;
- вкладені запити на одній сторінці виконуються паралельно;
- модульність, фільтри, gzip, byte-ranges (докачка), chunked відповіді, HTTP-аутентифікація, SSI-фільтр;
- акселероване проксіювання з підтримкою кешування;
- обслуговування статичних запитів, індексних файлів, автоматичне створення списку файлів, кеш дескрипторів відкритих файлів.

Конфігурація HTTP-сервера nginx дозволяє визначати віртуальні веб-сервери (директива server), які фізично знаходяться й обслуговуються одним сервером. Віртуальні сервери поділяються на локації (location). Для віртуального сервера можливо задати адреси і порти, на яких будуть прийматися з'єднання, а також імена, які можуть включати * для позначення довільній послідовності в перший і останній частині, або задаватися регулярним виразом. Локації можуть задаватися точним URI, частиною URI, або регулярним виразом. location'и

можуть бути налаштовані для обслуговування запитів зі статичного файлу, проксування на http, fastcgi чи memcached сервер.

5.4 Gunicorn

Gunicorn - це сервер HTTP-сервера шлюзу веб-сервера Python (WSGI). Це попередня робоча модель, перенесена з проекту Unicorn від Ruby. Сервер Gunicorn широко сумісний з низкою веб-фреймворків, просто реалізований, легкий на ресурсах сервера і досить швидкий.

5.5 Опис процесу впровадження програмного забезпечення

Нижче покроково наведено опис дій для впровадження програмного забезпечення на віртуальному приватному сервері (VPS) з операційною системою Ubuntu Server версії 16.04 та використанням технологій Docker, Nginx та Gunicorn:

1. Необхідно інсталювати Docker, web-сервер Gunicorn та проксі-сервер Nginx для подальшого запуску процесу впровадження системи;
2. В теці з вихідними файлами коду проекту програмного забезпечення необхідно створити файл налаштувань для Docker-контейнера який буде містить web-сервер з системою. Файл має назву «Dockerfile» та містить необхідні інструкції та вказівки за допомогою яких система буде впроваджена у виділеному та ізольованому Docker-контейнері;
3. В теці з вихідними файлами коду проекту програмного забезпечення необхідно створити файл налаштувань для Docker-контейнерів системи, бази даних, та проксі-серверу. Файл має назву «docker-compose.yml» та містить необхідні інструкції та вказівки за допомогою яких система, база даних та проксі-сервер будуть впроваджені у виділених та ізольованих Docker-контейнерах;

4. За допомогою команди «docker-compose build» запускається процес створення Docker-контейнерів використовуючи інструкції зазначені у файлах «Dockerfile» та «docker-compose.yml»;
5. По закінченню виконання попередньої команди необхідно ввести команду «docker-compose up» для підняття створених контейнерів.
6. Після закінчення виконання команди необхідно перейти до web-браузеру та у пошуковому рядку домене ім’я або IP-адресу, які призначені віртуальному приватному серверу, на якому впроваджена система, для перевірки справності роботи програмного забезпечення;
7. Після перевірки, необхідно закрити запущений процес, за допомогою якого було піднято контейнери (зазвичай комбінація клавіш «Ctrl+C»);
8. Для того, щоб система функціонувала у фоновому режимі необхідно підняти контейнери за допомогою команди «docker-compose up -d».

5.6 Висновки до розділу

В даному розділі було розглянуто шлях та засоби за допомогою яких дане програмне забезпечення (систему) можна впровадити. За допомогою таких технологій як Docker, Nginx та Gunicorn значно легше впровадити систему на будь-якій робочій машині. Незалежно, чи це робоча машина розробника чи спеціально відведений web-сервер.

ВИСНОВКИ

В процесі роботи над дипломним проектом проаналізовано проблеми оцінювань учнів та/або студентів під час проходження акредитованих паперових оцінювань на територіях навчальних закладів. Тоді було вирішено спроектувати систему (програмне забезпечення), яке би вирішувало проблеми несправедливої та нечесної оцінки здобутих учнями/студентами теоретичних та/або практичних знань.

Спроектоване програмне забезпечення (система) має ряд переваг перед вже існуючими рішеннями, а саме:

- система є єдиною для всіх навчальних закладів. Немає потреби впроваджувати систему ізольовано для кожного навчального закладу;
- кожен навчальний заклад проходить перевірку перед тим як бути частиною системи, тому буде відсутність неакредитованих оцінювань (тестів);
- є можливість перегляду статистики вчителями та викладачами здач, а також огляду власних результатів учнями та студентами;

Також проведено аналіз якості інформаційної системи, проведено модульне та ручне тестування, усунені знайдені недоліки.

Розроблено необхідну проектну документацію, схеми процесів застосунку, варіантів використань, а також керівництво з впровадження програного забезпечення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python 3.7.3 documentation - Офіційна документація мови програмування Python [Електронний ресурс] - Режим доступу: https://docs.python.org/3/ – Назва з екрану;
2. Django documentation – Офіціна документація фреймворку до мови програмування Python – Django [Електронний ресурс] - Режим доступу: https://docs.djangoproject.com/en/2.2/ – Назва з екрану;
3. PostgreSQL: Documentation – Офіційна документація системи керування базами даних PostgreSQL [Електронний ресурс] - Режим доступу: https://www.postgresql.org/docs/ – Назва з екрану;
4. Introduction to HTML – Сайт з вивчення мови HTML [Електронний ресурс] - Режим доступу: https://www.w3schools.com/html/html_intro.asp – Назва з екрану;
5. CSS Tutorial – Сайт з вивчення CSS [Електронний ресурс] - Режим доступу: https://www.w3schools.com/css/default.asp – Назва з екрану;
6. Introduction – Bootstrap – Офіційна документація набору інструментів і шаблонів Bootstrap [Електронний ресурс] - Режим доступу: https://getbootstrap.com/docs/4.3/getting-started/introduction/ – Назва з екрану;
7. ECMAScript Language Specification - ECMA-262 Edition 5.1 – Офіційна документація мови програмування JavaScript [Електронний ресурс] - Режим доступу: https://www.ecma-international.org/ecma-262/5.1/ – Назва з екрану;
8. NGINX Docs | Welcome to NGINX documentation – Офіційна документація проксі-серверу NGINX [Електронний ресурс] - Режим доступу: https://docs.nginx.com – Назва з екрану;
9. Docker Documentation – Офіційна документація інструменту керування Linux-контейнерами Docker [Електронний ресурс] - Режим доступу: https://docs.docker.com – Назва з екрану;

10. Gunicorn – WSGI server – Gunicorn 19.9.0 documentation – Офіційна документація web-серверу Gunicorn [Електронний ресурс] - Режим доступу: http://docs.gunicorn.org/en/stable/index.html - Назва з екрану;
11. Лучано Рамальо – «Fluent Python: Clear, Concise, and Effective Programming» - 2015 р. 750 ст.;
12. Антоніо Малє – «Django 2 by Example: Build powerful and reliable Python web applications from scratch» - 2018 р. 526 ст.;
13. Том Аратин- «Building Django 2.0 Web Applications Create enterprise-grade, scalable Python web applications easily with Django 2.0» - 2018 р. 397 ст.;
14. Джон Дакет – «HTML and CSS: Design and Build Websites» - 2011 р. 490 ст.;
15. Джон Дакет – «JavaScript and JQuery: Interactive Front-End Web Development» - 2014 р. 640 ст.;
16. Тести ЗНО онлайн [Електронний ресурс] – Режим доступу: https://zno.osvita.ua – Назва з екрану;
17. Online Test Pad - Online Test Pad - Онлайн тести, опитування, кросворди. Онлайн конструктор тестів, опитувань, кросвордів. Віджети для вашого сайту. [Електронний ресурс] – Режим доступу: https://onlinetestpad.com/ua - Назва з екрану;
18. Online Testing Free Quiz Maker Create the Best quizzes | ClassMarker [Електронний ресурс] – Режим доступу: https://www.classmarker.com – Назва з екрану.